Processing of Biological Data

Prof. Dr. Volkhard Helms
Tutor: Hanah Robertson
Winter Semester 2025-26

Saarland University
Chair for Computational Biology

Exercise Sheet 1

Due: 05.11.2025 10:00 am

Submission

- You are advised to work in groups of two people.
- Submit your solutions (with code listings) as a single PDF attachment to zbi.compbio.pobd@gmail.com. Please hand in your source code also via mail. Late submissions will not be considered.
- Do not forget to mention your names/matriculation numbers.
- You are free to use any programming language to solve the problems.

Principal Component Analysis (PCA) and Data Imputation

Exercise 1.1: Principal Component Analysis (40 points)

Write a program that applies the PCA technique to the toy dataset in the supplement ("pca_toy.txt").

(a) Standardize the variables in the dataset. Why is this step necessary before performing a PCA? (5 points)

(b) Create a scatter plot from the transformed data, with PC1 and PC2 as the axes. (10 points)

(c) Which variables are the most important for PC1 and PC2? Why? (10 points)

(d) What percentage of the variance in the dataset is explained by PC1 and PC2? (10 points)

(e) Under what circumstances would more PCs be considered? (5 points)

Hint: You can use the function ***prcomp*** *in R, or* ***sklearn.decomposition.PCA*** *in Python.*

Exercise 1.2: Data Imputation (60 points)

Data imputation techniques can be used to fill out missing data in sparse dataframes. Here, we will try to generate missing entries in a proteomics dataset ("ms_toy.txt") that were below the detection limit for expression data. Missing values are denoted as "NA".

1. Imputation based on a given data distribution (30 points)

First, we will write a function that randomly generates the missing data based on the distribution of the existing data. Given a variable, the function should:

1. Calculate the mean and the standard deviation
2. Derive a mean value for the distribution of the missing values from the lower quantile of the old distribution. The new mean should be in the lower quantile since we want to simulate the low expression data (see Figure 1).

   Hint: In R, this can be done with the function ***qnorm****.*
3. Derive the new standard deviation by taking a fraction of the overall standard deviation.

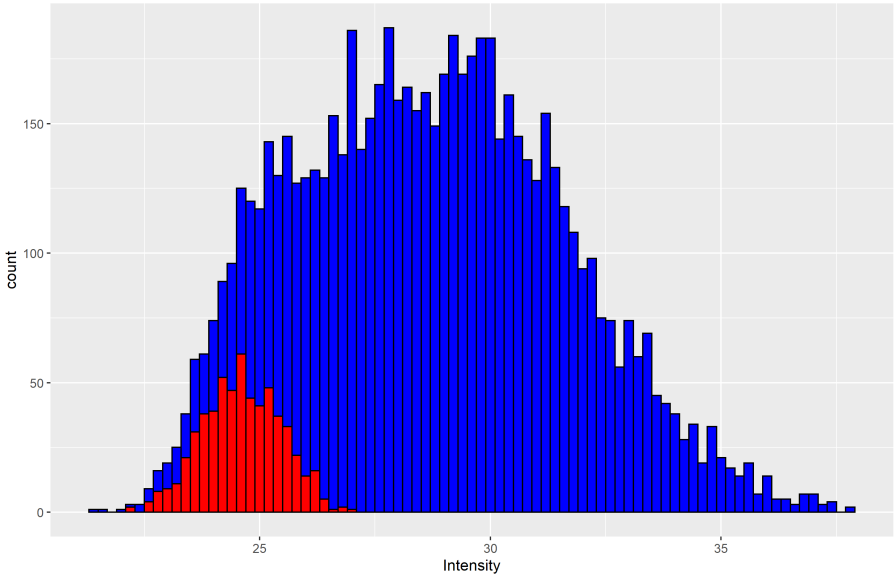

Figure 1: Histogram of *ctrl.1* data after imputation (blue=overall sample, red=imputed data)

4. Generate the new data by randomly drawing values from a normal distribution with the new mean and standard deviation from the two previous steps.

   Hint: In R, use the function ***rnorm****.*

Apply your function to the variable *ctrl.1* and plot the overall sample and the imputed data in a similar fashion to Figure 1.

- What is the effect of changing the percentile parameter in Step 2 and the ratio parameter in Step 3?
- Which configuration of parameters is the most logical/desirable?

2. k-Nearest Neighbor imputation (30 points)

Missing data in one variable can also be derived from other variables in the same dataset. One of these approaches is the *k-Nearest Neighbor* (kNN) imputation. Here, we will fill in missing values in the control samples (*ctr.1, ctrl.2, ctrl.3*).

1. Create a dataset containing only the three control samples (*ctr.1, ctrl.2, ctrl.3*).
2. Remove any row where all three samples have missing values (since there is no data to derive anything from).
3. Perform a k-Nearest Neighbor imputation on the dataset by using a library of your choice.

   Hint: In R, this is implemented as ***kNN*** *in the "VIM" package. In python,* ***KNNImputer*** *from sklearn.impute.*

Again, plot the imputed data for the variable *ctrl.1* in a similar way to Figure 1, and compare it to the plot from the first part of the exercise (e.g. mean, standard deviation, number of values). What are the advantages and disadvantages of the *kNN* method, compared to imputation based on the data distribution?